

# Trade Study of DEM Software

Thomas Stucky, Damiana Catanoso, Jason Fugate (KBR, Inc. at NASA Ames Research Center)

- [Review of Open-source Discrete Element Method Softwares](#)
  - [YADE Usage & Experience](#)
    - [Source: https://yade-dem.org/doc/GPUacceleration.html](https://yade-dem.org/doc/GPUacceleration.html)
  - [ESyS-Particle Usage & Experience](#)
  - [CHRONO Usage & Experience](#)
- [Simulator Comparisons](#)
  - [Simulation Parameters](#)
  - [Trade Study](#)
    - [Error Analysis](#)
      - [Error Compared to Testbed](#)
      - [Error compared to EDEM](#)
- [Conclusion and Software Selection](#)
  - [Run-time Force/Torque Computation](#)
  - [Yade Integration with OceanWATERS](#)
  - [Published Material Resulting from Study](#)
- Appendix A: Smooth vs. Non-Smooth Contact Monitoring
- Appendix B: Sample Collection Strategy
- Appendix C: Simulation Parameters for EDEM

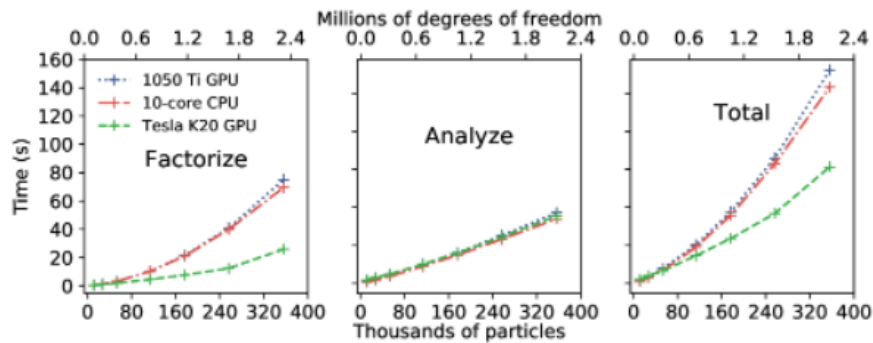
## Review of Open-source Discrete Element Method Software Packages

This is an internal trade study, authored as a Confluence wiki page. Three open-source solutions have been investigated for integration into OceanWATERS: Yade, EsysParticle, and ProjectChrono.

### YADE Usage & Experience

YADE, an open-source framework for discrete numerical models, focused on Discrete Element Method, is an option that was considered for OceanWATERS integration. The computational foundation of YADE is written in C++ using a flexible object model, allowing independent implementation of new algorithms, interfaces with other software packages (e.g. flow simulation), and data import/export routines. Python can be used to create and manipulate the simulation or for post-processing. The Python wrapper usage provides a convenient coding interface that makes the software relatively simple to implement. Getting started with YADE is best done with the utilization of its thorough tutorial program located at the following site: <https://yade-dem.org/doc/tutorial.html>. This resource provides a variety of different example use cases and simulation setups that can be a great starting point for a multitude of potential user applications that might differ from those developed by OceanWATERS.

Another benefit to YADE is its active user community. There are online forums on which users can submit questions and typically expect to receive feedback within 24-48 hours. This could also be helpful for users looking to adapt the OceanWATERS software for alternative use cases. YADE allows for particle bonding, complex polyhedra particle shapes, and multi-sphere particle configurations, which provides versatility in terrain specification for the user. The number of particles is limited only by the user's system processing capability. However, it is important to note that one of the limitations of YADE is that it is currently only shared-memory parallel, and as such, it will probably not scale to larger problem sizes that would require High Performance Computing (HPC) clusters. GPU/CPU computations are allowed in YADE; however, an in-depth evaluation of the benefits of GPU computing as opposed to standard CPU computing was abandoned due to existing documentation showing the benefits for an arbitrary simulation when the number of particles is varied. It can be seen that analytical computations are relatively unaffected by using GPU computing when compared to CPU computing. The benefit is shown in factorizing computations, and becomes particularly magnified as the particle number increases. It may be worthwhile to investigate for users depending on simulation size.



Source: <https://yade-dem.org/doc/GPUacceleration.html>

When compared to commercial EDEM solutions in sand, snow, and ice, YADE was able to generate comparable solutions. However, of the other open source frameworks used, YADE actually had the worst matching in the primary force component directions. Overall, the trend and magnitude of the datasets were within reason, and the solver makes up for much of these discrepancies through its overall functionality, which is the most ideal of the options considered. It is also worth noting that despite the discrepancies with the commercial solution, YADE actually provided the best matching when compared to the experimental testbed solution for sand in the primary force component directions. The simulation time was found to be approximately equal to the other open-source and commercial solution tools for a matching experiment.

## ESyS-Particle Usage & Experience

ESyS-Particle, another of the open-source DEM frameworks considered for OceanWATERS implementation, was specifically designed for execution on parallel supercomputers, clusters or multi-core PCs running a Linux(or Windows)-based operating system. This differs from YADE and presents the ability to scale up to very large simulations requiring potentially millions of particles, which is the main benefit for ESyS. ESyS has been demonstrated to scale (weakly) to greater than 30,000 CPU cores. The C++ simulation engine for ESyS implements spatial domain decomposition via the Message Passing Interface (MPI). A Python wrapper API provides flexibility in the design of numerical models, specification of modeling parameters and contact logic, and analysis of simulation data. Similar to YADE, ESyS therefore allows the user to operate through a relatively convenient coding language with a wide range of functionality. ESyS-Particle has been used to simulate earthquake nucleation, comminution in shear cells, silo flow, rock fragmentation, and fault gouge evolution, to name but a few applications beyond the OceanWATERS use case. It has been actively developed since 1994, when it originated at The Centre for Geoscience Computing at the University of Queensland, Brisbane, Australia. It is a fully featured framework including, an MPI parallel simulation engine, a Python API for simulation setup and execution, a scriptable setup of model geometry, non-rotational and rotational spherical particles, triangular meshes for specifying boundary conditions and walls, a scriptable visualization of particle assemblies utilizing POVray and VTK, and a variety of particle-particle and particle-wall interaction laws, such as:

- linear elastic repulsion between un-bonded contacting particles
- linear elastic bonds between bonded particle-pairs
- both non-rotational and rotational frictional interactions between un-bonded particles
- rotational bonds implementing torsion and bending stiffnesses in addition to normal and shear stiffnesses

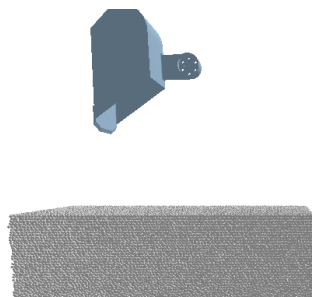
ESyS also has a tutorial with very thorough documentation located here: [https://launchpadlibrarian.net/187793286/ESyS-Particle\\_Tutorial.pdf](https://launchpadlibrarian.net/187793286/ESyS-Particle_Tutorial.pdf); however, it is 153 pages long, and can be very difficult to follow as opposed to the succinct documentation and examples that YADE has. It might be a bit overwhelming for a user to attempt to sort through the documentation and find information about how to apply the software for their specific use case if it differs from the OceanWATERS setup. The ESyS wiki page, as listed on

their website, appears to be shut down (at least temporarily). Alternatively, there is a nice forum for ESyS users where answers to questions and solutions to problems are provided typically within a 24-48 hour period. There don't appear to be quite as many users of the software as there are for YADE, but there are a multitude of resources to rely upon as a user. ESyS does not allow for polyhedral particle shapes, but with the ability to clump together spherical particles, users still have a reasonable degree of adaptability for terrain modeling. One limitation of ESyS is that it currently does not allow for GPU computing, so although a user can scale to an HPC, a single user may be unable to significantly reduce simulation time from its baseline.

As for the data, ESyS also compared quite well in magnitude and trend to the commercial solver EDEM. In fact, it provided the best overall matching in the primary force directions compared to the other open-source solutions. It did not perform quite as well when compared to the experimental testbed, but it is still viewed as a quality solution tool for our purposes including slightly fewer key features when compared to YADE. The simulation time was found to be more or less equal to the other open-source and commercial solution tools for a matching experiment.

## CHRONO Usage & Experience

ProjectCHRONO (Chrono for short) is considered a multi-physics simulation engine, which means it is capable of a multitude of physics models, including: multibody dynamics, finite element, mechatronics, robotics, and terramechanics. Chrono actually seems to have multiple modules geared towards terramechanics simulations. Chrono::Parallel implements most of Chrono's core physics models (both non-smooth and smooth contact dynamics) in a parallel CPU processing framework, which allows for faster simulation for scenarios incorporating thousands of interacting elements. Chrono::Granular seeks to do the same thing for a GPU framework; however, it remains in early development and presently has far fewer options implemented, which is why we rated Chrono's GPU support as "partial". Chrono::Vehicle applies Chrono's core physics models in ways that assist in vehicle simulations, which include wheels rolling over terrain.



Animated GIF compiled from POV files generated by a Chrono simulation.

The simulations we ran in Chrono to evaluate it for OceanWATERS utilized Chrono::Vehicle's GranularTerrain class integrated with Chrono::Parallel's ChSystemParallelNSC class. ChSystemParallelNSC is a parallel CPU capable version of Chrono's core non-smooth contact system, and GranularTerrain is a physics system wrapper that allows for easier placement of spherical particles within a 3D rectangular region and a function that removes particles from behind a followed object (i.e. a wheel or scoop) and places them in front thereby lowering total particles and computation time.

The major differences that set Chrono apart from the other DEM simulators listed here are:

- The non-smooth contact model is different from other discrete element methods. It does allow time steps to be larger; however, total computation time still winds up being about the same as for other simulators since each time step also takes longer to compute. To learn more about the differences between the smooth and non-smooth contact modeling see [Appendix A - Smooth vs. Non-Smooth Contact Monitoring](#).
- Compliance is used for material elastic properties instead of Youngs modulus and Poisson ratio. We were unable to calculate the compliance from the Youngs modulus and Poisson ratio in our particular case; however, tests were done to vary this value across multiple orders of magnitude that are typical of compliance for other materials, and the results showed no appreciable change in force behavior or magnitude for the trenching simulation from the default compliance value of 0. Therefore, it was considered accurate enough to leave this parameter at its default.
  - This may imply that elastic properties of a material in general do not have a large affect on the forces produced during trenching and that other properties such as particle shape, mass, and friction are of greater importance. This would inform our development strategy going forward; however, a more thorough investigation would need to be done to confirm this. In particular, trying the same test on another DEM simulator.

Since Chrono is so large and complex, there were multiple paths available to simulate trenching. There were other methods tried, and they are listed here:

- The same trench simulation done with Chrono's non-smooth contact model (NSC) was attempted using its smooth contact model (SMC) to see if computation was faster or if more accurate results were produced. Sadly the time step required for an SMC parallel computing system (ChSystemParallelSMC) to remain stable were so small that the entire trenching simulation would have taken weeks, and results were never produced. It is possible that SMC in parallel mode works better for larger numbers of particles, but the parallelizing overhead is too large for a performance boosts at the numbers of particles we used.
- The trench simulation was also attempted with Chrono::Granular, which utilizes GPU computation. Being in early development, the API is limited in its options. It at first had difficulty importing our scoop mesh, but that was solved by reexporting the scoop mesh from Blender (which presumably fixed some error in it that somehow wasn't triggering problems in other import code). Now the scoop mesh imports, but there are "outside of domain" error being generated.

## Simulator Comparisons

Three different simulation scenarios have been implemented and ran in each candidate. The three scenarios involve the first pass of the sample collection strategy described in [Appendix B - Sample Collection Strategy](#).

Scenario	Pass	Material	Bite Depth (mm)
A	1st	snow	15
B	1st	sand	3
C	1st	ice	15

## Simulation Parameters

The simulation parameters implemented in each simulation vary. The following table compares those parameters side-by-side.

Parameter	EDEM	ProjectChrono (NSC)	ProjectChrono (SMC)	ProjectChrono (GPU-SMC)	Yade	ESyS-Particle
Rolling friction	yes	yes	yes	yes	yes	yes

Static friction	yes	yes	yes	yes	yes	yes
Sliding friction	yes	yes	no	no	no	no
Spinning friction	yes	yes	no	no	no	no
Young's Modulus	yes	no	yes	no	yes	yes
Shear modulus	yes	no	no	no	yes	no
Poisson Ratio	yes	no	yes	no	yes	no
Restitution Coefficient	yes	yes	yes	no	yes	yes
Cohesion	no	yes	yes (labeled "adhesion")	yes	yes	yes
Compliance/Stiffness	no?	yes (normal, tangential, rolling, spinning)	no	yes	yes	yes

**yes** - The simulator implements this quantity as an adjustable parameter

**no** - It does implement it as an adjustable parameter.

Specific parameters and scoop trajectory details for each of the 3 scenarios are provided in [Appendix C - Simulation Parameters for EDEM](#)

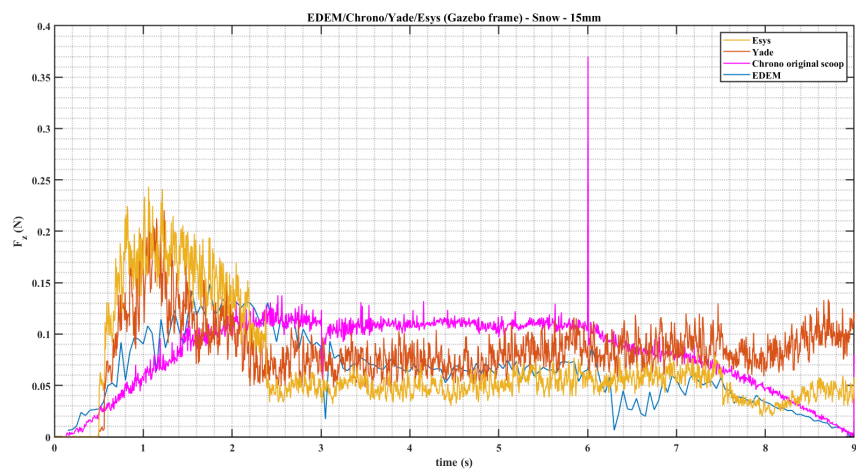
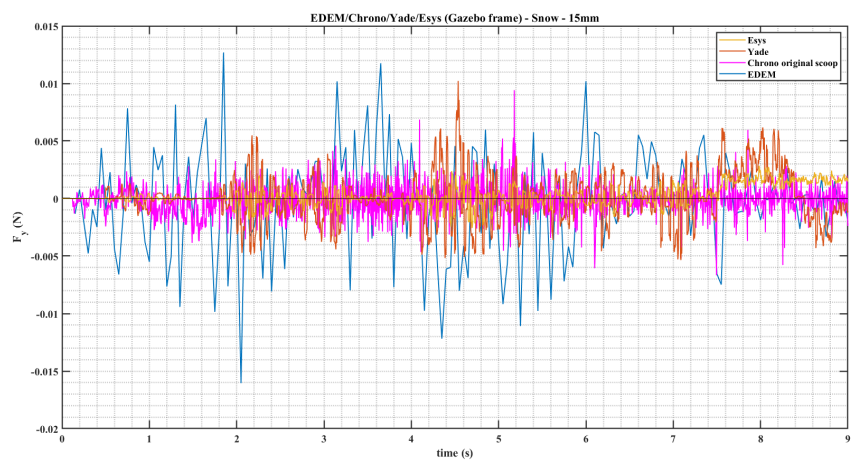
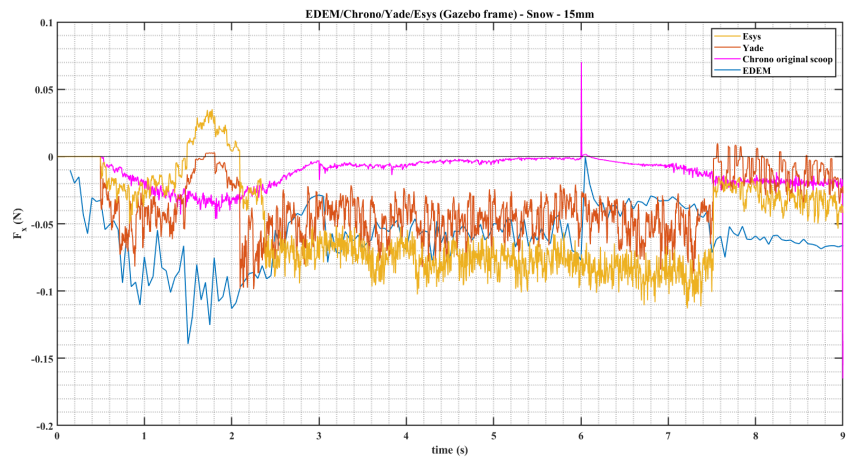
## Trade Study

The trade study was performed by creating a list with all the parameters that reflect OceanWATERS' requirements, assigning points (on a scale of 0 to 1) to each candidate for each parameter, and performing a weighted sum of the points to have a final score which will determine the choice of the DEM solution to integrate in OceanWATERS. What follows are the results of that evaluation.

	Scripting wrappers	C++ API	Limit in number of particles	Particle bonding	Polyhedral particle shape	Multi-sphere particle	Parallel computations	Super-computer suitable	Clear documentation	Active community [0, 10]	GPU capable	Fx normalized Score Compared to EDEM	Fy normalized Score Compared to EDEM	Fz normalized Score Compared to EDEM	Fx normalized Score Compared to Testbed	Fy normalized Score Compared to Testbed	Fz normalized Score Compared to Testbed	TOTAL
EDEM	1	0	0	1	0	1	1	0	1	1	1	1	1	1	0.68	1	0	0.84
YADE	1	1	0	1	1	1	1	0	1	0.8	1	0	0.36	0	1	0	0.42	0.68
ESYS PARTICLE	1	1	0	1	0	1	1	1	0.5	0.6	0	1	0	0.68	0	0.25	1	0.67
CHRONO	1	1	0	0	1	0	1	1	1	0.7	0.5	0.56	1	1	0.23	0.92	0.77	0.60
Weight	0.65	0	0	1	0.5	0.8	0	0.4	0.6	0.75	0.5	0.9	0.2	0.7	0.45	0.05	0.15	
Notes	Questions on next steps: 1) licensing for Yade? 2) tasks assignment for what to do next (thursday meeting)? 3) planning for integration? 4) be sure we include computation time in the report.																	

Screen capture of the trade study evaluation table created by the team in Google Sheets.

The final 6 columns come from comparing the open source data to the same scenarios run in EDEM. Results for sand have been compared to experimental data from the physical testbed (see above) as well. The figures below show comparison plots for the three scenarios.



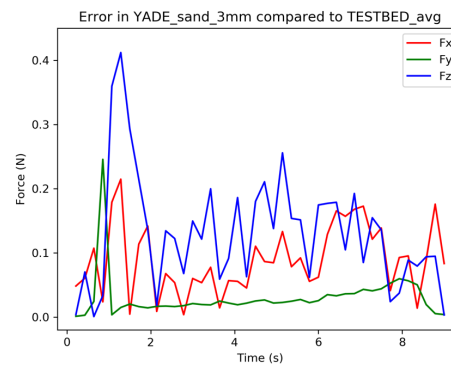
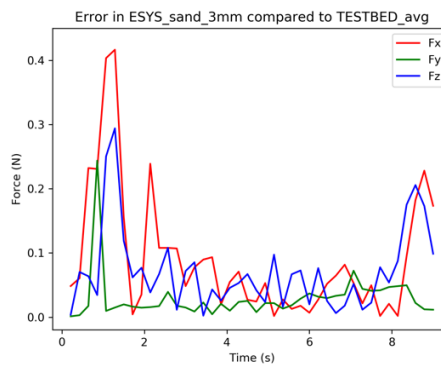
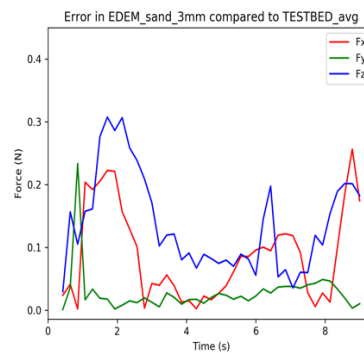
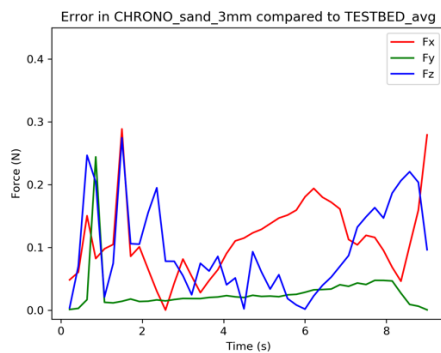
## Error Analysis

The normalized root-mean-squared deviation (NRMSD) for each force component (Fx, Fy, Fz) and for each simulator (EDEM, YADE, ESys, and Chrono) was compared to the "ideal" simulation to numerically determine which simulation data came out closest to the desired result. In our case we considered two different "ideal" simulations: EDEM and a physical testbed at NASA Ames, the former being a very well validated commercial discrete element software, and the latter being a real-world validation testbed that collects wrench data acting on a 3D printed replica of the scoop on the Mars Phoenix lander as it is pushed through sand.

For the evaluation table we needed a 0-1 score, where 1 represents the best possible result, so the normalized root-mean-squared deviation was used to calculate what we are calling the "Normalized Comparison Score". Due to systemic errors we are aware of in the physical testbed, the normalized comparison scores for the testbed were underweighted compared to the EDEM scores.

### Error Compared to Testbed

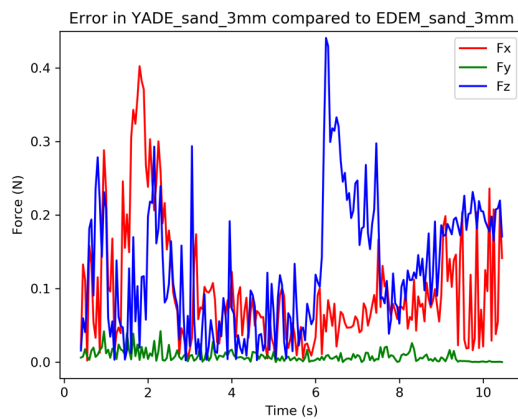
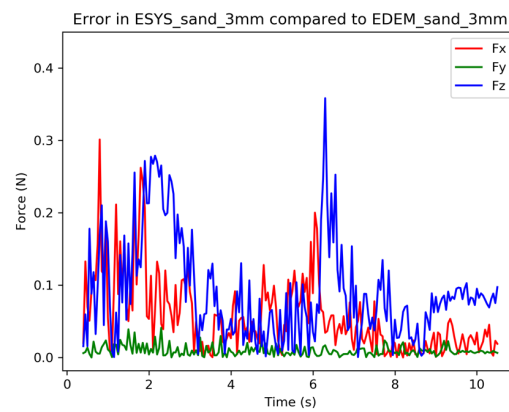
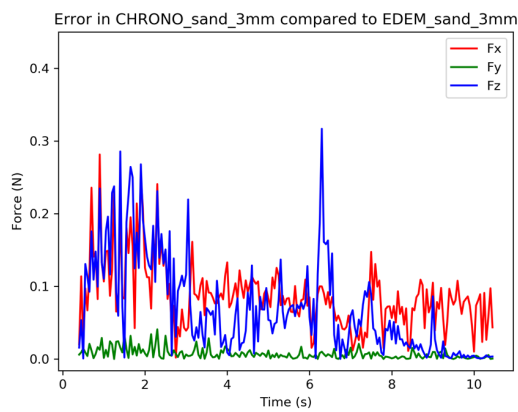
Simulator	NRMSD	NRMSD	NRMSD
	Fx	Fy	Fz
ProjectCHRONO	50%	19%	41%
ESys	52%	19%	33%
YADE	41%	19%	54%
EDEM	44%	19%	70%





## Error compared to EDEM

Simulator	NRMSD	NRMSD	NRMSD
	F <sub>x</sub>	F <sub>y</sub>	F <sub>z</sub>
ProjectCHRONO	22%	13%	15%
ESys	17%	15%	18%
YADE	28%	14%	24%



## Conclusion and Software Selection

As previously mentioned, an evaluation based on points assignment and weighted sum was carried out. First, a list of important aspects with regards to the DEM software implementation in OceanWATERS was made. Each candidate was graded with a number of points on a 0 to 1 scale. Then, each category was assigned with a weight, on a 0 to 1 scale, representing the importance of that specific aspect. Finally, a weighted average was calculated for each DEM candidate. As a result, Yade scored 0.68, EsysParticle 0.67, Chrono 0.60. This suggests that Yade and Esys Particle are the most suitable candidates. Because of its ease of use and large online community and documentation, Yade was chosen as best candidate for implementation in OceanWATERS.



## Run-time Force/Torque Computation

Computation of force and torque resulting from a custom terrain interaction scenario may be supported in OceanWATERS at a later software release. This potential feature would allow users to specify:

- Material properties of the scooped terrain, such as: density, internal friction, cohesion, and elastic properties.
- End effector properties, such as: shape, mass, trajectory, and the friction coefficient with the terrain.
- Simulation parameters typical of DEM, such as: particle size and time step interval.

This could be accomplished by integrating the open source DEM software Yade into OceanWATERS.

Users interested in this potential feature should be aware that computations of this nature are much slower than real-time. Depending on the material and end effector properties and the simulation parameters selected, a custom terrain interaction scenario can take up to 20 hours or more to process on a modern laptop or workstation. Users interested in eventually incorporating this potential feature into their studies should carefully consider how long each simulation would run and schedule accordingly.

## Yade Integration with OceanWATERS

The Yade integration can be accomplished with writing a Cosimulation Gazebo plugin. The plugin would allow the user to initially either select from preset terrain profiles, such as snow, sand, and ice, which have been used and tested in development, or to provide characterization parameters for a custom terrain of their own. As previously mentioned, the user-specified parameters may include material properties, end-effector properties, and/or YADE simulation parameters. Additionally, the user may specify where raw force and torque feedback output files are generated. This provides the user with high levels of flexibility in catering the simulation to a specific use case. The parameters would be utilized to create a wrapper for the use case that interacts with the Yade API to generate a simulation. The terrain interaction scenario would be completed by Yade and the output files would be found in the directory specified by the user.

## Published Material Resulting from Study

Our intern, Damiana Catanoso, took it upon herself to submit a conference paper to IEEE about the trade study.

D. Catanoso, T. Stucky, J. Case and A. Rogg, "Analysis of Sample Acquisition Dynamics Using Discrete Element Method," *2020 IEEE Aerospace Conference*, 2020, pp. 1-11, doi: [10.1109/AERO47225.2020.9172431](https://doi.org/10.1109/AERO47225.2020.9172431).

## Appendix A - Smooth vs. Non-Smooth Contact Monitoring

### Smooth

- Penalty methods, contacts are deformable
- More time
- Smooth dynamics: velocity and position smooth functions of time
- Chrono's approach for contacts: assumption of perfectly rigid body (elastic). contacts lead to complementarity constraints. Complementarity problem= $\max/\min$  of function of two vectors subject to complementarity constraint=they have to be orthogonal (inner product=0).
- DAE solver can solve only smooth.

### Non-smooth

- Complementary solver in case of contact
- Non smooth dynamics
- Fast, large time-step
- Nonsmooth dynamics: Doesn't require position and velocity functions in time to be smooth anymore. Smooth function=continuous, with existing derivatives. During impact velocity is not continuous, function non smooth. Two particles colliding are a nonsmooth dynamical system. Smooth dynamics get to same result using infinitely long transients to arrive at equilibrium position. Smooth dynamics for rapidly changing velocity has steep gradients, so high computation time. Approach: model fast phenomena as instantaneous, jumps between continuous functions. For modeling interaction between rigid bodies: nonsmooth force laws for contact, impact, friction
- Chrono Approach. DVI time stepping method valid for also nonsmooth. There are other solvers in chrono but only DVI can handle non-smooth. Dynamic model expressed as a MDI. write the model in discrete form, iterative solution across timestep. Impulse-based method. All contact forces, whether finite or instantaneously infinite are treated uniformly, as impulses accruing over a single time step.

### References

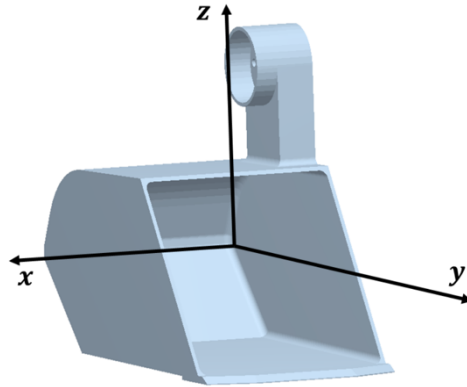
[http://www.projectchrono.org/assets/white\\_papers/ChronoCore/integrator.pdf](http://www.projectchrono.org/assets/white_papers/ChronoCore/integrator.pdf)

<https://www.osti.gov/servlets/purl/751199>

## Appendix B - Sample Collection Strategy

### Scoop Model

A simplified scoop, based on the one used on the Phoenix Mars Lander, was used for sample collection. The simplified scoop and its body frame are shown in Figure 1.



**Figure 1.** The scoop's body frame is centered on the center of mass of the scoop and having the X-axis parallel to the joint rotation axis, the Z-axis parallel and opposite to the gravity vector and the Y-axis obtained through the product  $Z \times X$ .

### Sample Collection Strategy

A sample collection strategy was developed to use in the OceanWATERS Gazebo framework and for future use on a physical lander. It is a simple strategy that suits the project needs; however, a specific study focused on optimizing the trajectory is yet to be conducted, so it is unlikely this will be the exact strategy employed in the actual Europa Lander mission.

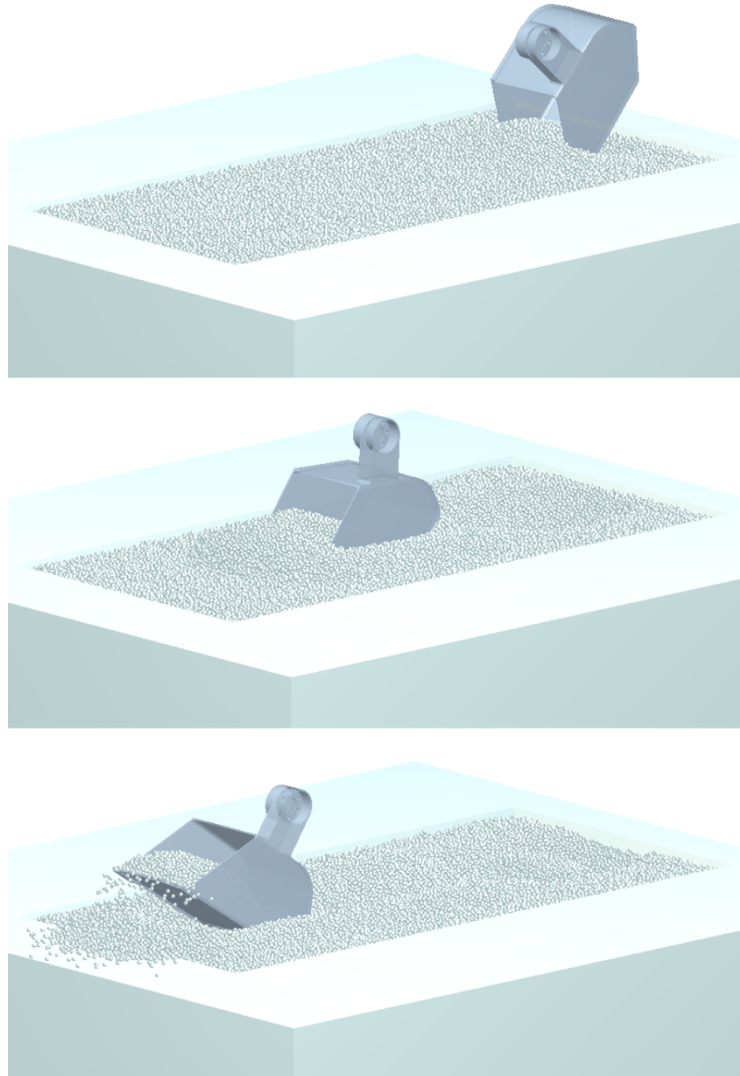
### Sample Passes

The sample collection strategy consists of a sequence of five passes of the scoop through the same terrain region digging deeper with each pass. When one pass is over, the scoop empties its collected sample into a safe zone away from the trenching area and starts on the next pass. Between two consecutive passes, the excavated depth increases a fixed quantity that we call bite depth. At the end of the scooping maneuver the total excavated depth will then be  $5 \times$  bite depth. This number was arbitrarily defined as a good trade off between number of simulations and trench final depth.

### Scoop Trajectory

All passes are characterized by the same scoop trajectory. **Figure 2** shows the three main steps constituting the scoop motion which are:

1. The scoop approaches and enters the soil through a rotation around its joint axis which ends with the scoop's blade parallel to the ground plane.
2. The scoop starts a linear motion parallel to the surface. It traverses a few scoop lengths.
3. The scoop performs a second rotation which ends the pass.



**Figure 2.** Three frame of the EDEM simulation. The top image shows the scoop entering the particle bed. Middle image shows the scoop traversing parallel to the surface. Bottom image shows the scoop exiting the granular material.

## Appendix C - Simulation Parameters for EDEM

### Scoop Parameters

These parameters were used for the scoop in all 3 simulation scenarios.

Parameter	Value used in EDEM
Scoop: material	Aluminum
Scoop: density	2.7g/cm3
Scoop: young's modulus	69 Gpa
Scoop: Poisson's ratio	0.33
Scoop-snow: Rolling friction	0.05
Scoop-snow: Static friction	0.58
Scoop-snow: Restitution	0.7

### Scenario A - Snow, Depth 15mm, Pass 1

#### Parameters

Parameter	Value
Bulk Material	snow
Snow-snow: Rolling friction	0.05
Snow-snow: Static friction	0.58
Snow-snow: Sliding friction	Not specified
Snow-snow: Spinning friction	Not specified
Snow: Youngs Modulus	2.72e+7 Pa
Snow: Shear modulus	10000000 Pa
Snow: Poisson Ratio	0.36
Snow-snow: Restitution	0.88
Snow-snow: Cohesion	No cohesion
Compliance/Stiffness	?
Snow: density	0.3 g/cm3
Snow-snow: Interaction Model	Elastic, non-cohesive
Gravity	1.3 m/s2

### Scoop Motion and Particle Specification

Parameter	Value
Pass number	1
Bite depth	15mm
Total depth*	15mm
Rotational velocity of scoop during circular motion	30 deg/s
Rotational displacement during each circular phase	90 deg
Linear velocity of scoop during linear motion	0.1 m/s
linear displacement during linear phase	0.3 m
Distance between Rotational axis and lower surface of scoop**	116.2 mm
Distance between Rotational axis and upper surface of scoop***	40.77 mm
Particle Sizes Used	3.5mm diameter
Particle Shapes	sphere

\*This is the distance between the bottom blade of the scoop and the surface level, when the scoop is in its linear motion

\*\*The rotation axis passes by the center of the circle

\*\*\*In physical testbed this is: 120 mm

## Scenario B - Sand, 3mm Depth, Pass 1

### Parameters

Parameter	Value
Bulk Material	Sand
Sand-sand: Rolling friction	0.05
Sand-sand: Static friction	0.35
Sand-sand: Sliding friction	Not specified
Sand-sand: Spinning friction	Not specified
Sand: Youngs Modulus	2.5e+07 Pa
Sand: Shear modulus	10000000 Pa
Sand: Poisson Ratio	0.25
Sand-sand: Restitution	0.88
Sand-sand: Cohesion	No cohesion
Compliance/Stiffness	?
Sand: density	1.6 g/cm3

<b>Snow-snow: Interaction Model</b>	Elastic, non-cohesive
<b>Gravity on Europa</b>	1.3 m/s <sup>2</sup>

## Scoop Motion and Particle Specification

**PLEASE NOTE:** Values here are same as snow except for depth. Also for physical testbed the position of the axis of rotation is different than simulation.

Parameter	Value
<b>Pass number</b>	1
<b>Bite depth</b>	3mm
<b>Total depth</b>	3mm
<b>Rotational velocity of scoop during circular motion</b>	30 deg/s
<b>Rotational displacement during each circular phase</b>	90 deg
<b>Linear velocity of scoop during linear motion</b>	0.1 m/s
<b>linear displacement during linear phase</b>	0.3 m
<b>Distance between Rotational axis and lower surface of scoop (see pics below)</b>	116.2 mm
<b>Distance between Rotational axis and upper surface of scoop (see pics below)</b>	40.77 mm
<b>Sphere diameter</b>	3.5mm diameter
<b>Particle Shape</b>	Single sphere

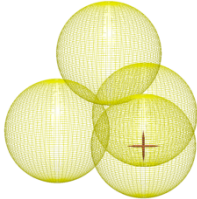
## Scenario C - Ice, 15mm Depth, Pass 1

### Parameters

Parameter	Value
<b>Bulk Material</b>	ice
<b>Ice-Ice: Rolling friction</b>	0.05
<b>Ice-Ice: Static friction</b>	0.58
<b>Ice-Ice: Sliding friction</b>	Not specified
<b>Ice-Ice: Spinning friction</b>	Not specified
<b>Ice: Youngs Modulus</b>	2.72e+7 Pa
<b>Ice: Shear modulus</b>	10000000 Pa
<b>Ice: Poisson Ratio</b>	0.36
<b>Ice-Ice: Restitution</b>	0.88
<b>Ice-Ice: Cohesion</b>	No cohesion



<b>Compliance/Stiffness</b>	?
<b>Ice: density</b>	0.9 g/cm <sup>3</sup>
<b>Ice-Ice: Interaction Model</b>	Elastic, non-cohesive
<b>Gravity on Europa</b>	1.3 m/s <sup>2</sup>

<b>Parameter</b>	<b>Value</b>
<b>Pass number</b>	1
<b>Bite depth</b>	15mm
<b>Total depth</b>	15mm
<b>Rotational velocity of scoop during circular motion</b>	30 deg/s
<b>Rotational displacement during each circular phase</b>	90 deg
<b>Linear velocity of scoop during linear motion</b>	0.1 m/s
<b>linear displacement during linear phase</b>	0.3 m
<b>Distance between Rotational axis and lower surface of scoop (see pics below)</b>	116.2 mm
<b>Distance between Rotational axis and upper surface of scoop (see pics below)</b>	40.77 mm
<b>Sphere diameter</b>	3.5mm diameter
<b>Particle Shape</b>	 <p>Tetrahedron: four sphere configuration</p>

### Scoop Motion and Particle Specification

**PLEASE NOTE:** Values here are same as snow except for particle shape.